Assignment 3
RTL Design

due
Friday, October 16 1998
12:30 PM

Question 1

In this assignment you will design, program and test a simple computer. The computer's CPU has two registers: a program counter (PC) and an accumulator (A). The CPU uses a Harvard architecture (separate data and instruction memories). The CPU has eight instructions. The data and instruction memories are both 8 bits wide and 32 words deep. Each instruction memory location holds a 3-bit opcode in the MS 3 bits and a 5-bit address field in the LS 5 bits.

The computer is divided into five parts:

- a read/write data memory
- a read-only instruction memory
- the PC datapath
- the A datapath
- a controller (instruction decoder)

Data Types

Create data types for instruction and data words. Each should be 8 bits wide. The data words should be subtypes of unsigned, the instructions should be subtypes of std_logic_vector. Put the type definitions in a package.

Memories

Define entities for the data and instruction memories. The instruction memory is a ROM with a 5-bit address input and an 8-bit output (the instruction). Use a selected assignment statement to implement the ROM. Note that you will not have to define the contents of all locations (see below).

The data memory is a RAM with the following inputs: "data memory in", address, read/write control signal, and clock. The only output is a "data memory out" output. Use an array of data words and the VHDL array indexing operator (( )) to implement the RAM.

A Datapath

The accumulator (A) datapath updates the accumulator as instructed by the controller. A can be loaded with: (1) A, (2) the data memory output, (3) the address field of the current instruction, (4) the result of an 'add', 'not', or 'and' operation on the data memory output and A.

The ALU datapath inputs are: the data memory output, the address field of the current instruction, an operation code, and the clock.

The outputs are: the data memory input, a signal that indicates A is zero, and a signal that indicates the MS bit of A is set (A is negative).

PC Datapath

The program counter (PC) datapath updates PC as instructed by the controller. PC can be loaded with: (1) PC, (2) PC+1 (to point to the next instruction), (3) the current instruction's address field (to branch to another instruction), or (4) zero (to reset the computer).

The PC datapath inputs are: the address field of the current instruction, an operation code, a reset signal, and the clock.

The output is: the current value of the program counter.

Instruction Decoder (Controller)

The instruction decoder uses the current instruction to control the A and PC datapaths.

It's inputs are: the MS 3 bits of the instruction memory output (the current instruction), and the zero and negative status signals from the ALU.

It's outputs are: the operation code to the A datapath, the operation code to the PC datapath, and the RAM read/write control signal.

Design the controller so that the CPU can execute the following instructions:

	code	description
LOAD	000	load - load A from the data memory location addressed by the instruction address field
STORE	001	store - store A in the data memory location addressed by the instruction address field
LOADI	010	load immediate - load A with the current instruction's address field (the MS 3 bits are cleared)
ADD	011	add - load A with the sum of A and the data memory location addressed by the instruction address field
NOT	100	not - load A with the one's complement of A
AND	101	and - load A with the AND of A and the data memory location addressed by the instruction address field
JZ	110	jump on zero - if A is zero load PC with the current instruction's address field
JN	111	jump on negative - if A is negative load PC with the current instruction's address field

Note that the instruction's address field is used to address data memory, address instruction memory, or as a source of immediate data depending on the instruction.

Assignment

Write VHDL descriptions of each of the five parts.

The instruction memory should contain the following program which initializes three memory locations, executes a loop three times and then goes into an infinite loop.

```
Address Opcode  Operand Comment
0       LOADI   0       ; set location 0 to zero
1       STORE   0
4       NOT             ; set location 1 to -1
5       STORE   1
2       LOADI   1       ; set location 2 to 1
3       STORE   2
6       LOADI   3       ; load 3
7       ADD     1       ; subtract 1
8       JNZ     7       ; loop until zero
9       JZ      9       ; infinite loop
```

Assemble the above program into the CPU's machine language and use it to design the instruction memory. Unused locations should be set to zero.

Test each part of your design *separately*. As a minimum, ensure the following:

- the instruction memory gives the correct output for each address input
- if you write the values 1, `0FFH` `0EEH` to memory locations 0, 1 and 31 you can read back the same values from these memory location.
- the PC is properly reset, loaded, and incremented
- the ALU performs each of its six operations correctly
- the controller generates the correct (sequence of) outputs for each instruction, including the two possible outputs for each conditional jump instruction

Write a top-level entity that combines the five components. This entity should have two inputs: reset and clock, and three outputs: PC, the instruction memory output, A. Simulate the operation of your computer from reset until it executes two iterations of the infinite loop (11 instructions).

Submit the VHDL source code listings and simulation outputs. Bonus marks will be awarded for the simplest and for the easiest-to-read solutions.